WHY "COMPUTING" REQUIRES SYMBOLS

Zenon Pylyshyn
University of Western Ontario

For some years now I have been advocating the view that to understand what is essential about cognition as computing it is mandatory that we preserve a number of distinctions. I have discussed several of these distinctions in my book (Pylyshyn, 1984). For the present purpose I wish to examine one of these distinctions: that between a machine and the symbolically encoded "rules and representations" that the machine uses. (It doesn°t matter here whether by "the machine" one means the device described in the manufacturer°s manual, or what is sometimes called the "virtual machine" consisting of the raw machine plus an interpreter for some higher level programming language. This is just a conceptual distinction in any case since the virtual machine is no less a real physical machine than the one delivered from the manufacturer, only with a different initial state.) Since the distinction between the machine and the symbol structures is one of those distinctions that some people have been trying to do away with (cf., Anderson and Hinton, 1981), I will review one of the fundamental reasons why I believe that the task of providing explanations in cognitive psychology cannot be carried out successfully without it.

The difference between a very complicated device that goes through distinguishable states (but is not characterized as processing symbols) and what I would call a computer in the strict sense (as well as in the usual computer science sense) is exactly the difference between a Turing Machine and any arbitrarily complicated finite state automaton, network, or "connectionist" machine. The main difference, from our perspective, is not that the Turing Machine°s tape is unbounded (though that does have consequences whose relevance to cognitive science is not clear), but that when we do not impose a bound as part of the definition of the machine itself we force a certain kind of qualitative organization on the system. In particular it forces us to distinguish between a strictly finite mechanism (the Turing machine°s finite state "control box") and a finite but unbounded string of symbols. If it were not for that distinction it would not be possible to have a Universal Turing machine. The finite characterization of machines that such a distinction gives us is crucial. Turing machines are individuated by their finite part -- that°s what allows them to be enumerated. A finite part is similarly required for proof theory (the axioms and rules of inference have to be finitely specified).

It is important to see that what is at stake here is the nature of the organization captured in a certain description. An ordinary Von Neumann style computer can clearly be characterized as a finite state automaton. It can also be given a true description at the circuit level. But it°s only when it is described as processing symbols (and in fact only when it°s viewed as processing the particular symbols that are semantically interpreted) that we can explain its input-output behavior in such a way as to capture those regularities that are invariant over certain implementation differences. And what°s even more to the point, it°s only when we describe it at the symbol level that we can explain what it°s doing in semantic terms (e.g., in terms of doing arithmetic, or playing chess, or carrying out inferences, or whatever else the device may be correctly described as

doing). That (at least some) human reasoning (e.g., doing arithmetic, deciding what to have for dinner, planning a trip, deciding on the intended referent of an anaphoric expression, etc.) is correctly characterized in terms of such rules cannot be in dispute. The only arguable point has been whether the behavior described by such rules can be realized by a system that works according to some principles that do not reflect the structure of these rules.

Consider a simple example. A semantically interpretable procedure such as one for adding two numbers cannot be adequately described in terms of state-transition diagrams, such as those used in the description of finite state automata. The reason is that the general rule for adding numbers cannot be finitely stated as a rule for producing a transition from state Sn to state Sn+1 in a computer. Rather, it must be stated as a rule (or a set of rules) for transforming an expression of numerals into a new expression. An interpreted rule, such as the rule for addition, applies to states which have a particular semantic interpretation (say, as certain numbers).

Of course changes in the machine°s state are the result of physical, not number-theoretic, causes. Consequently the way the machine must work in order to be correctly described as following an interpreted (e.g., mathematical) rule, is that on every occasion in which the rule is invoked there must be physical properties of the machine°s state that are capable of serving as physical codes for that semantic interpretation. In other words, for each distinct rule-relevant semantic property there must be a corresponding distinct physical property associated with that state: distinct semantic properties must be preserved by some distinct physical properties -- and in fact they must be the very same physical properties that cause the machine to behave as it does on that occasion. Such articulation of the states into distinct properties must, furthermore, correspond to the articulation of the semantic rule in terms of symbolic expressions. In other words, the articulation of the states must be made explicit if we are to both express the rules that govern the computation and at the same time show how, in principle, such rules might be realized in a physical system.

There are all standard ideas. What they come down to is that in order to finitely express some computational regularity, such as that captured by a mathematical (or other) rule, we have to refer to a structure of symbols, and the structure of the expressions must be preserved by the structure of the states of the system. A characterization of the machinery that does not articulate the states of the system in this way cannot explain how the system can exhibit regularities expressed in the form of such rules as rules of inference. Thus if human behavior can be correctly described as following rules -- if capturing important regularities requires such a formulation -- then it appears that this has implications for the nature of the system that realizes such behavior.

People who object to the conventional view of computation as symbol processing frequently have in mind the implausibility of the mind working like a VAX. I have much sympathy for that view, as I keep saying: that°s why it°s so important in cognitive science to find out what the functional architecture of the mind is. I would not be the least surprised to find that it is so very different from a Von Neumann machine that it may scarcely

be recognizeable as a computer by examining its command set. It will, no doubt have massive parallelism. Many people think that having a lot of parallelism will make a fundamental difference to what we count as computing. But the issue is not whether the mind is a serial or a highly parallel computer. The issue is whether it processes symbols: whether it has rules and representations. A highly parallel system can process symbols in at least two ways. One is that it may be parallel only in the way it implements its primitive functions, i.e., the functional architecture may be neurally implemented in a highly parallel way. But, of course, that much is true of the Von Neumann computer. Its random access memory mechanism requires a great deal of simultaneous activity in every part of the memory. The other way that it may be parallel is that the primitive operations need not form a total ordering in time. But even in an architecture as radically nonlinear as one based on populations of ACTORS, there is no conflict with the sense of computing that I claim must be going on in the mind, so long as each actor processes semantically interpreted symbols, as opposed to just sending activations that have no semantic interpretations in the domain of our perceptions, thoughts, and the like.

The point of all this is to suggest that so long as cognition (human or otherwise) involves semantic regularities, such as knowledge based decisions and inferences, and so long as we view it as computing in any sense, we will need to view it as computing over symbols. No connectionist device, however complex, will do. Nor will any analog computer, but that is a topic for another occasion (for example, see Pylyshyn, 1984).

References

Anderson, J.A., and Hinton, G.E. "Models of Information Processing in the Brain," in G.E. Hinton & J.A. Anderson (Ed) Parallel Models of Information Processing in the Brain. Hillsdale, NJ: Erlbaum, 1981.

Pylyshyn, Z.W. Computation and Cognition: Toward a Foundation for Cognitive Science. Cambridge, Mass: M.I.T. Press (Bradford), 1984.